SEQ3: Differentiable Sequence-to-Sequence-to-Sequence Autoencoder for Unsupervised Abstractive Sentence Compression

SEQ^3: Differentiable Sequence-to-Sequence-to-Sequence Autoencoder for Unsupervised Abstractive Sentence Compression

Christos Baziotis[1,2], **Ion Androutsopoulos**[2], **Ioannis Konstas**[3], **Alexandros Potamianos**[1]
[1] School of ECE, National Technical University of Athens, Athens, Greece
[2] Department of Informatics, Athens University of Economics and Business, Athens, Greece
[3] Interaction Lab, School of Math. and Comp. Sciences, Heriot-Watt University, Edinburgh, UK
cbaziotis@mail.ntua.gr, ion@aueb.gr
i.konstas@hw.ac.uk, potam@central.ntua.gr

Abstract

Neural sequence-to-sequence models are currently the dominant approach in several natural language processing tasks, but require large parallel corpora. We present a sequence-to-sequence-to-sequence autoencoder (SEQ^3), consisting of two chained encoder-decoder pairs, with words used as a sequence of discrete latent variables. We apply the proposed model to unsupervised abstractive sentence compression, where the first and last sequences are the input and reconstructed sentences, respectively, while the middle sequence is the compressed sentence. Constraining the length of the latent word sequences forces the model to distill important information from the input. A pretrained language model, acting as a prior over the latent sequences, encourages the compressed sentences to be human-readable. Continuous relaxations enable us to sample from categorical distributions, allowing gradient-based optimization, unlike alternatives that rely on reinforcement learning. The proposed model does not require parallel text-summary pairs, achieving promising results in unsupervised sentence compression on benchmark datasets.

1 Introduction

Neural sequence-to-sequence models (SEQ2SEQ) perform impressively well in several natural language processing tasks, such as machine translation (Sutskever et al., 2014; Bahdanau et al., 2015) or syntactic constituency parsing (Vinyals et al., 2015). However, they require massive parallel training datasets (Koehn and Knowles, 2017). Consequently there has been extensive work on utilizing non-parallel corpora to boost the performance of SEQ2SEQ models (Sennrich et al., 2016; Gülçehre et al., 2015), mostly in neural machine translation where models that require absolutely no parallel corpora have also been proposed (Artetxe et al., 2018; Lample et al., 2018b).

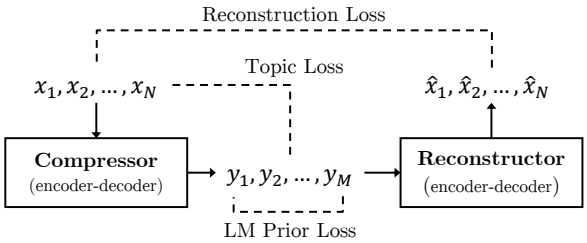

Figure 1: Overview of the proposed SEQ^3 autoencoder.

Unsupervised (or semi-supervised) SEQ2SEQ models have also been proposed for summarization tasks with no (or small) parallel text-summary sets, including unsupervised sentence compression. Current models, however, barely reach lead-N baselines (Fevry and Phang, 2018; Wang and Lee, 2018), and/or are non-differentiable (Wang and Lee, 2018; Miao and Blunsom, 2016), thus relying on reinforcement learning, which is unstable and inefficient. By contrast, we propose a sequence-to-sequence-to-sequence autoencoder, dubbed SEQ^3, that can be trained end-to-end via gradient-based optimization. SEQ^3 employs differentiable approximations for sampling from categorical distributions (Maddison et al., 2017; Jang et al., 2017), which have been shown to outperform reinforcement learning (Havrylov and Titov, 2017). Therefore it is a generic framework which can be easily extended to other tasks, e.g., machine translation and semantic parsing via task-specific losses. In this work, as a first step, we apply SEQ^3 to unsupervised abstractive sentence compression.

SEQ^3 (§2) comprises two attentional encoder-decoder (Bahdanau et al., 2015) pairs (Fig. 1): a compressor C and a reconstructor R. C (§2.1) receives an input text $\mathbf{x} = \langle x_1, \ldots, x_N \rangle$ of N words, and generates a summary $\mathbf{y} = \langle y_1, \ldots, y_M \rangle$ of M words (M<N), $\mathbf{y}$ being a latent variable. R and C communicate only through the discrete words of the summary $\mathbf{y}$ (§2.2). R (§2.3) produces a sequence $\hat{\mathbf{x}} = \langle \hat{x}_1, \ldots, \hat{x}_N \rangle$ of N words from $\mathbf{y}$, try-

Proceedings of NAACL-HLT 2019, pages 673–681
Minneapolis, Minnesota, June 2 - June 7, 2019.

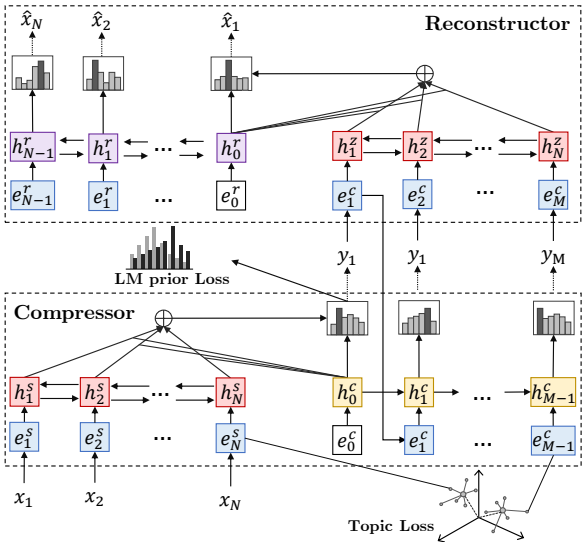

Figure 2: More detailed illustration of SEQ^3. The compressor (C) produces a summary from the input text, and the reconstructor (R) tries to reproduce the input from the summary. R and C comprise an attentional encoder-decoder each, and communicate only through the (discrete) words of the summary. The LM prior incentivizes C to produce human-readable summaries, while topic loss rewards summaries with similar topic-indicating words as the input text.

ing to minimize a reconstruction loss $L_R = (\mathbf{x}, \hat{\mathbf{x}})$ (§2.5). A pretrained language model acts as a prior on $\mathbf{y}$, introducing an additional loss $L_P(\mathbf{x}, \mathbf{y})$ that encourages SEQ^3 to produce human-readable summaries. A third loss $L_T(\mathbf{x}, \mathbf{y})$ rewards summaries $\mathbf{y}$ with similar topic-indicating words as $\mathbf{x}$. Experiments (§3) on the Gigaword sentence compression dataset (Rush et al., 2015) and the DUC-2003 and DUC-2004 shared tasks (Over et al., 2007) produce promising results.

Our contributions are: (1) a fully differentiable sequence-to-sequence-to-sequence (SEQ^3) autoencoder that can be trained without parallel data via gradient optimization; (2) an application of SEQ^3 to unsupervised abstractive sentence compression, with additional task-specific loss functions; (3) state of the art performance in unsupervised abstractive sentence compression. This work is a step towards exploring the potential of SEQ^3 in other tasks, such as machine translation.

2 Proposed Model

2.1 Compressor

The bottom left part of Fig. 2 illustrates the internals of the compressor C. An embedding layer projects the source sequence $\mathbf{x}$ to the word embeddings $\mathbf{e}^s = \langle e^s_1, \ldots, e^s_N \rangle$, which are then encoded by a bidirectional RNN, producing $\mathbf{h}^s = \langle h^s_1, \ldots, h^s_N \rangle$. Each h^s_t is the concatenation of the corresponding left-to-right and right-to-left states (outputs in LSTMs) of the bi-RNN.

$$h^s_t = [\overrightarrow{\text{RNN}}_s(e^s_t, \overrightarrow{h}^s_{t-1}); \overleftarrow{\text{RNN}}_s(e^s_t, \overleftarrow{h}^s_{t+1})]$$

To generate the summary $\mathbf{y}$, we employ the attentional RNN decoder of Luong et al. (2015), with their global attention and input feeding. Concretely, at each timestep ($t \in \{1, \ldots, M\}$) we compute a probability distribution a_i over all the states $h^s_1, \ldots, h^s_N$ of the source encoder conditioned on the current state h^c_t of the compressor's decoder to produce a context vector c_t.

$$a_i = \text{softmax}(h^{s\intercal}_i W_a h^c_t), \quad c_t = \sum_{i=1}^{N} a_i h^s_i$$

The matrix W_a is learned. We obtain a probability distribution for y_t over the vocabulary V by combining c_t and the current state h^c_t of the decoder.

$$o^c_t = \tanh(W_o [c_t; h^c_t] + b_o) \quad (1)$$

$$u^c_t = W_v o^c_t + b_v \quad (2)$$

$$p(y_t | y_{<t}, \mathbf{x}) = \text{softmax}(u^c_t) \quad (3)$$

W_o, b_o, W_v, b_v are learned. c_t is also used when updating the state h^c_t of the decoder, along with the embedding e^c_t of y_t and a countdown argument $M - t$ (scaled by a learnable w_d) indicating the number of the remaining words of the summary (Fevry and Phang, 2018; Kikuchi et al., 2016).

$$h^c_{t+1} = \overrightarrow{\text{RNN}}_c(h^c_t, e^c_t, c_t, w_d\,(M - t)) \quad (4)$$

For each input $\mathbf{x} = \langle x_1, \ldots, x_N \rangle$, we obtain a target length M for the summary $\mathbf{y} = \langle y_1, \ldots, y_M \rangle$ by sampling (and rounding) from a uniform distribution $U(\alpha N, \beta N)$; α, β are hyper-parameters ($\alpha < \beta < 1$); we set $M = 5$, if the sampled M is smaller. Sampling M, instead of using a static compression ratio, allows us to train a model capable of producing summaries with varying (e.g., user-specified) compression ratios. Controlling the output length in encoder-decoder architectures has been explored in machine translation (Kikuchi et al., 2016) and summarization (Fan et al., 2018).

2.2 Differentiable Word Sampling

To generate the summary, we need to sample its words y_t from the categorical distributions $p(y_t | y_{<t}, \mathbf{x})$, which is a non-differentiable process.

Soft-Argmax Instead of sampling y_t, a simple workaround during training is to pass as input to the next timestep of C's decoder and to the corresponding timestep of R's encoder a weighted sum of all the vocabulary's (V) word embeddings, using a peaked softmax function (Goyal et al., 2017):

$$\overline{e}_t^c = \sum_i^{|V|} e(w_i)\,\text{softmax}(u_t^c/\tau) \quad (5)$$

where u_t^c is the unnormalized score in Eq. 2 (i.e., the logit) of each word w_i and $\tau \in (0, \infty)$ is the temperature. As $\tau \to 0$ most of the probability mass in Eq. 5 goes to the most probable word, hence the operation approaches the $\arg\max$.

Gumbel-Softmax We still want to be able to perform sampling, though, as it has the benefit of adding stochasticity and facilitating exploration of the parameter space. Hence, we use the Gumbel-Softmax (GS) reparametrization trick (Maddison et al., 2017; Jang et al., 2017) as a low variance approximation of sampling from categorical distributions. Sampling a specific word y_t from the softmax (Eq. 3) is equivalent to adding (element-wise) to the logits an independent noise sample ξ from the Gumbel distribution[1] and taking the $\arg\max$:

$$y_t \sim \text{softmax}(u_t^c) \leftrightarrow y_t = \arg\max(u_t^c + \xi) \quad (6)$$

Therefore, using the GS trick, Eq. 5 becomes:

$$\tilde{e}_t^c = \sum_i^{|V|} e(w_i)\,\text{softmax}((u_t^c + \xi)/\tau) \quad (7)$$

Straight-Through Both relaxations lead to mixtures of embeddings, which do not correspond to actual words. Even though this enables the compressor to communicate with the reconstructor using continuous values, thus fully utilizing the available embedding space, ultimately our aim is to constrain them to communicate using *only* natural language. In addition, an unwanted discrepancy is created between training (continuous embeddings) and test time (discrete embeddings). We alleviate these problems with the Straight-Through estimator (ST) (Bengio et al., 2013). Specifically, in the forward pass of training we discretize $\tilde{e}_t^c$ by using the $\arg\max$ (Eq. 6), whereas in the backward pass we compute the gradients using the GS (Eq. 7). This is a biased estimator due to the mismatch between the forward and backward passes, but works well in practice. ST GS reportedly outperforms scheduled sampling (Goyal et al., 2017) and converges faster than reinforcement learning (Havrylov and Titov, 2017).

[1] $\xi_i = -\log(-\log(x_i)),\ \ x_i \sim U(0,1)$

2.3 Reconstructor

The reconstructor (upper right of Fig. 2) works like the compressor, but its encoder operates on the embeddings $e_1^c, \ldots, e_M^c$ of the words $y_1, \ldots, y_M$ of the summary (exact embeddings of the sampled words y_t in the forward pass, approximate differentiable embeddings in the backward pass).

2.4 Decoder Initialization

We initialize the hidden state of each decoder using a transformation of the concatenation $[\overrightarrow{h_N^s}; \overleftarrow{h_1^s}]$ of the last hidden states (from the two directions) of its bidirectional encoder and a length vector, following Mallinson et al. (2018). The length vector for the decoder of the compressor C consists of the target summary length M, scaled by a learnable parameter w_v, and the compression ratio $\frac{M}{N}$.

$$h_0^c = \tanh(W_c\,[\overrightarrow{h_N^s}; \overleftarrow{h_1^s}; w_v M; \frac{M}{N}])$$

W_c is a trainable hidden layer. The decoder of the reconstructor R is initialized similarly.

2.5 Loss Functions

Reconstruction Loss $L_R(\mathbf{x}, \hat{\mathbf{x}})$ is the (negative) log-likelihood assigned by the (decoder of) R to the input (correctly reconstructed) words $\mathbf{x} = \langle x_1, \ldots, x_N \rangle$, where p_R is the distribution of R.

$$L_R(\mathbf{x}, \hat{\mathbf{x}}) = -\sum_{i=1}^{N} \log p_R(\hat{x}_i = x_i)$$

We do not expect $L_R(\mathbf{x}, \hat{\mathbf{x}})$ to decrease to zero, as there is information loss through the compression. However, we expect it to drive the compressor to produce such sentences that will increase the likelihood of the target words in the reconstruction.

LM Prior Loss To ensure that the summaries $\mathbf{y}$ are readable, we pretrain an RNN language model (see Appendix) on the *source* texts of the full training set. We compute the Kullback-Leibler divergence D_{KL} between the probability distributions of the (decoder of) the compressor ($p(y_t|y_{<t}, \mathbf{x})$, Eq. 3) and the language model ($p_{\text{LM}}(y_t|y_{<t}, \mathbf{x})$). Similar priors have been used in sentence compression (Miao and Blunsom, 2016) and agent communication (Havrylov and Titov, 2017).

We also use the following task-specific losses.

Topic Loss Words with high TF-IDF scores are indicative of the topic of a text (Ramos et al., 2003; Erkan and Radev, 2004). To encourage the compressor to preserve in the summary **y** the topic-indicating words of the input **x**, we compute the TF-IDF-weighted average v^x of the word embeddings of **x** and the average v^y of the word embeddings of **y** and use their cosine distance as an additional loss $L_T = 1 - \cos(v^x, v^y)$.

$$v^x = \sum_{i=1}^{N} \frac{\text{IDF}(x_i)\, e_i^s}{\sum_{t=1}^{N} \text{IDF}(x_t)} \qquad v^y = \frac{1}{M} \sum_{i=1}^{M} e_i^c$$

(Using TF-IDF in v^y did not help.) All IDF scores are computed on the training set.

Length Penalty A fourth loss L_L (not shown in Fig. 1) helps the (decoder of the) compressor to predict the end-of-sequence (EOS) token at the target summary length M. L_L is the cross-entropy between the distributions $p(y_t|y_{<t}, \mathbf{x})$ (Eq. 3) of the compressor at $t = M + 1$ and onward, with the one-hot distribution of the EOS token.

2.6 Modeling Details

Parameter Sharing We tie the weights of layers encoding similar information, to reduce the number of trainable parameters. First, we use a shared embedding layer for the encoders and decoders, initialized with 100-dimensional GloVe embeddings (Pennington et al., 2014). Additionally, we tie the shared embedding layer with the output layers of both decoders (Press and Wolf, 2017; Inan et al., 2017). Finally, we tie the encoders of the compressor and reconstructor (see Appendix).

OOVs Out-of-vocabulary words are handled as in Fevry and Phang (2018) (see Appendix).

3 Experiments

Datasets We train SEQ^3 on the *Gigaword* sentence compression dataset (Rush et al., 2015).[2] It consists of pairs, each containing the first sentence of a news article (**x**) and the article's headline (**y**), a total of 3.8M/189k/1951 train/dev/test pairs. We also test (without retraining) SEQ^3 on DUC-2003 and DUC-2004 shared tasks (Over et al., 2007), containing 624/500 news articles each, paired with 4 reference summaries capped at 75 bytes.

Methods compared We evaluated SEQ^3 and an ablated version of SEQ^3. We only used the article sentences (sources) of the training pairs from Gigaword to train SEQ^3; our model is *never* exposed to target headlines (summaries) during training or evaluation, i.e., it is completely unsupervised. Our code is publicly available.[3]

We compare SEQ^3 to other unsupervised sentence compression models. We note that the extractive model of Miao and Blunsom (2016) relies on a pre-trained attention model using at least 500K parallel sentences, which is crucial to mitigate the inefficiency of sampling-based variational inference and REINFORCE. Therefore it is not comparable, as it is semi-supervised. The results of the extractive model of Fevry and Phang (2018) are also not comparable, as they were obtained on a different, not publicly available test set. We note, however, that they report that their system performs worse than the LEAD-8 baseline in ROUGE-2 and ROUGE-L on Gigaword. The only directly comparable unsupervised model is the abstractive 'Pretrained Generator' of Wang and Lee (2018). The version of 'Adversarial REINFORCE' that Wang and Lee (2018) consider unsupervised is actually weakly supervised, since its discriminator was exposed to the summaries of the same sources the rest of the model was trained on.

As baselines, we use LEAD-8 for Gigaword, which simply selects the first 8 words of the source, and PREFIX for DUC, which includes the first 75 bytes of the source article. We also compare to supervised abstractive sentence compression methods (Tables 1-3). Following previous work, we report the average F1 of ROUGE-1, ROUGE-2, ROUGE-L (Lin, 2004). We implemented SEQ^3 with LSTMs (see Appendix) and during inference we perform greedy-sampling.

Results Table 1 reports the Gigaword results. SEQ^3 outperforms the unsupervised Pretrained Generator across all metrics by a large margin. It also surpasses LEAD-8. If we remove the LM prior, performance drops, esp. in ROUGE-2 and ROUGE-L. This makes sense, since the pretrained LM rewards correct word order. We also tried removing the topic loss, but the model failed to converge and results were extremely poor (Table 1). Topic loss acts as a bootstrap mechanism, biasing the compressor to generate words that maintain the topic of the input text. This greatly reduces variance due to sampling in early stages of training, alleviating the need to pretrain individual

[2] github.com/harvardnlp/sent-summary

[3] https://github.com/cbaziotis/seq3

Type	Supervision	Methods	R-1	R-2	R-L
Supervised	3.8M	ABS (Rush et al., 2015)	29.55	11.32	26.42
		SEASS (Zhou et al., 2017)	36.15	17.54	33.63
		words-lvt5k-1sent (Nallapati et al., 2016)	**36.4**	**17.7**	**33.71**
Weakly supervised	(3.8M)	Adversarial REINFORCE (Wang and Lee, 2018)	28.11	9.97	25.41
Unsupervised	0	LEAD-8 (Baseline)	21.86	7.66	20.45
		Pretrained Generator (Wang and Lee, 2018)	21.26	5.60	18.89
		SEQ^3 (Full)	**25.39**	**8.21**	**22.68**
		SEQ^3 w/o LM prior loss	24.48	6.68	21.79
		SEQ^3 w/o TOPIC loss	3.89	0.1	3.75

Table 1: Average results on the (English) Gigaword dataset for abstractive sentence compression methods.

Model	R-1	R-2	R-L
ABS (Rush et al., 2015)	28.48	8.91	23.97
PREFIX	**21.3**	**6.38**	**18.82**
SEQ^3 (Full)	20.90	6.08	18.55

Table 2: Averaged results on the DUC-2003 dataset; the top part reports results of supervised systems.

Model	R-1	R-2	R-L
TOPIARY (Zajic et al., 2007)	25.12	6.46	20.12
Woodsend et al. (2010)	22	6	17
ABS (Rush et al., 2015)	28.18	8.49	23.81
PREFIX	20.91	5.52	18.20
SEQ^3 (Full)	**22.13**	**6.18**	**19.3**

Table 3: Averaged results on the DUC-2004 dataset; the top part reports results of supervised systems.

components, unlike works that rely on reinforcement learning (Miao and Blunsom, 2016; Wang and Lee, 2018). Overall, both losses work in synergy, with the topic loss driving *what* and the LM prior loss driving *how* words should be included in the summary. SEQ^3 behaves similarly on DUC-2003 and DUC-2004 (Tables 2-3), although it was trained on Gigaword. In DUC-2003, however, it does not surpass the PREFIX baseline.

Finally, Fig. 3 illustrates three randomly sampled outputs of SEQ^3 on Gigaword. In the first one, SEQ^3 copies several words esp. from the beginning of the input (hence the high ROUGE-L) exhibiting extractive capabilities, though still being adequately abstractive (bold words denote paraphrases). In the second one, SEQ^3 showcases its true abstractive power by paraphrasing and compressing multi-word expressions to single content words more heavily, still without losing the overall meaning. In the last example, SEQ^3 progressively becomes ungrammatical though interestingly retaining some content words from the input.

input: the american sailors who **thwarted** somali pirates flew home to the u.s. on wednesday but without their captain , who was still aboard a navy destroyer after being rescued from the **hijackers** .
gold: us sailors who thwarted pirate hijackers fly home
SEQ^3: the american sailors who **foiled** somali pirates flew home after crew **hijacked** .

input: the central election commission -lrb- cec -rrb- on monday **decided that taiwan will hold another election** of national assembly members on may # .
gold: national <unk> election scheduled for may
SEQ^3: the central election commission -lrb- cec UNK **announced elections** .

input: dave bassett resigned as manager of struggling english premier league side nottingham forest on saturday after they were knocked out of the f.a. cup in the third round , according to local reports on saturday .
gold: forest manager bassett quits .
SEQ^3: dave bassett resigned as manager of struggling english premier league side UNK forest on knocked round press

Figure 3: Good/bad example summaries on Gigaword.

4 Limitations and Future Work

The model tends to copy the first words of the input sentence in the compressed text (Fig. 3). We hypothesize that since the reconstructor is autoregressive, i.e., each word is conditioned on the previous one, errors occurring early in the generated sequence have cascading effects. This inevitably encourages the compressor to select the first words of the input. A possible workaround might be to modify SEQ^3 so that the first encoder-decoder pair would turn the inputs to longer sequences, and the second encoder-decoder would compress them trying to reconstruct the original inputs. In future work, we plan to explore the potential of SEQ^3 in other tasks, such as unsupervised machine translation (Lample et al., 2018a; Artetxe et al., 2018) and caption generation (Xu et al., 2015).

Acknowledgments

We would like to thank Ryan McDonald for helpful discussions and feedback. This work has been partially supported by computational time granted from the Greek Research & Technology Network (GR-NET) in the National HPC facility - ARIS. We thank NVIDIA for donating a TitanX GPU.

References

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018. Unsupervised neural machine translation. In *Proceedings of the Annual Meeting of International Conference on Learning Representations*.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*, San Diego, California.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of the SIGNLL Conference on Computational Natural Language Learning*, pages 10–21, Berlin, Germany. Association for Computational Linguistics.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the Conference of the NAACL:HLT*, pages 93–98, San Diego, California.

Günes Erkan and Dragomir R. Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

Angela Fan, David Grangier, and Michael Auli. 2018. Controllable abstractive summarization. In *Proceedings of the Workshop on Neural Machine Translation and Generation*, pages 45–54, Melbourne, Australia.

Thibault Fevry and Jason Phang. 2018. Unsupervised sentence compression using denoising auto-encoders. In *Proceedings of the Conference on Computational Natural Language Learning*, pages 413–422, Brussels, Belgium.

Kartik Goyal, Chris Dyer, and Taylor Berg-Kirkpatrick. 2017. Differentiable scheduled sampling for credit assignment. In *Proceedings of the Annual Meeting of the ACL*, pages 366–371, Vancouver, Canada.

Jiatao Gu, Daniel Jiwoong Im, and Victor O. K. Li. 2018. Neural machine translation with gumbel-greedy decoding. In *Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5125–5132.

Caglar Gulcehre, Sarath Chandar, and Yoshua Bengio. 2017. Memory augmented neural networks with wormhole connections. *arXiv preprint arXiv:1701.08718*.

Çaglar Gülçehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loïc Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. *arXiv preprint arXiv:1503.03535*.

Serhii Havrylov and Ivan Titov. 2017. Emergence of language with multi-agent games: Learning to communicate with sequences of symbols. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Proceedings of the Advances in Neural Information Processing Systems*, pages 2149–2159.

Sepp Hochreiter and Jrgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying word vectors and word classifiers: A loss framework for language modeling. *CoRR*, abs/1611.01462.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparametrization with gumble-softmax. In *Proceedings of the International Conference on Learning Representations*, Toulon, France.

Yuta Kikuchi, Graham Neubig, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2016. Controlling output length in neural encoder-decoders. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1328–1338, Austin, Texas.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*.

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations*, Banff, AB, Canada.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. *arXiv preprint arXiv:1706.03872*.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, Hervé Jégou, et al. 2018a. Word translation without parallel data. In *Proceedings of the International Conference on Learning Representations*, Vancouver, Canada.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018b. Unsupervised machine translation using monolingual corpora only. In *Proceedings of the International Conference on Learning Representations*, Vancouver, Canada.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The concrete distribution: A continuous relaxation of discrete random variables. In *Proceedings of the International Conference on Learning Representations*, Toulon, France.

Jonathan Mallinson, Rico Sennrich, and Mirella Lapata. 2018. Sentence compression for arbitrary languages via multilingual pivoting. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2453–2464, Brussels, Belgium.

Yishu Miao and Phil Blunsom. 2016. Language as a Latent Variable: Discrete Generative Models for Sentence Compression. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 319–328, Austin, Texas.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of the Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany.

Paul Over, Hoa Dang, and Donna Harman. 2007. Duc in context. *Information Processing and Management*, 43(6):1506–1520.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, Doha, Qatar.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics*, pages 157–163, Valencia, Spain. Association for Computational Linguistics.

Juan Ramos et al. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the 1st International Conference on Machine Learning*, volume 242, pages 133–142.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the Annual Meeting of the ACL*, pages 1073–1083, Vancouver, Canada.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the Annual Meeting of the ACL*, pages 86–96, Berlin, Germany.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 3104–3112.

Oriol Vinyals, Ł ukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 2773–2781.

Yaushian Wang and Hung-yi Lee. 2018. Learning to encode text as human-readable summaries using generative adversarial networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 4187–4195, Brussels, Belgium.

Kristian Woodsend, Yansong Feng, and Mirella Lapata. 2010. Title generation with quasi-synchronous grammar. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 513–523, Cambridge, MA.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the International Conference on Machine Learning*, volume 37, pages 2048–2057, Lille, France. PMLR.

David Zajic, Bonnie J. Dorr, Jimmy Lin, and Richard Schwartz. 2007. Multi-candidate reduction: Sentence compression as a tool for document summarization tasks. *Information Processing Management Special Issue on Summarization*, 43(6):1549–1570.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2017. Selective encoding for abstractive sentence summarization. In *Proceedings of the Annual Meeting of the ACL*, pages 1095–1104, Vancouver, Canada.

A Appendix

A.1 Temperature for Gumbel-Softmax

Even though the value of the temperature τ does not affect the forward pass, it greatly affects the gradient computation and therefore the learning process. Jang et al. (2017) propose to anneal τ during training towards zero. Gulcehre et al. (2017) propose to learn τ as a function of the compressor's decoder state h_t^c, in order to reduce hyper-parameter tuning:

$$\tau(h_t^c) = \frac{1}{\log(1 + \exp(w_\tau^\intercal h_t^c)) + 1} \quad (8)$$

where w_τ is a trainable parameter and $\tau(h_t^c) \in (0, 1)$. Havrylov and Titov (2017) add τ_0 as a hyper-parameter which controls the upper bound of the temperature.

$$\tau(h_t^c) = \frac{1}{\log(1 + \exp(w_\tau^\intercal h_t^c)) + \tau_0} \quad (9)$$

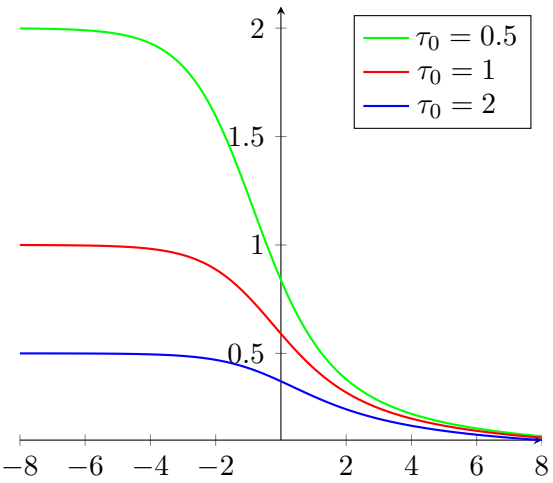

Figure 4: Plot of Eq. 9, with different values for the upper bound τ_0.

In our experiments, we had convergence problems with the learned temperature technique. We found that the compressor preferred values close to the upper bound, which led to unstable training, forcing us to set $\tau_0 > 1$ to stabilize the training process. Our findings align with the behavior reported by Gu et al. (2018). Consequently, we follow their choice and fix $\tau = 0.5$, which worked well in practice.

A.2 Out of Vocabulary (OOV) Words

The vocabulary of our experiments comprises the 15k most frequent words of Gigaword's training input texts (without looking at their summaries). To handle OOVs, we adopt the approach of Fevry and Phang (2018), which can be thought of as a simpler form of copying compared to pointer networks (See et al., 2017). We use a small set (10 in our experiments) of special OOV tokens OOV_1, OOV_2, ..., OOV_{10}, whose embeddings are updated during learning. Given an input text $\mathbf{x} = \langle x_1, \ldots, x_N \rangle$, we replace (before feeding $\mathbf{x}$ to SEQ^3) each unknown word x_i with the first unused (for the particular $\mathbf{x}$) OOV token, taking care to use the same OOV token for all the occurrences of the same unknown word in $\mathbf{x}$. For example, if 'John' and 'Rome' are not in the vocabulary, then "John arrived in Rome yesterday. While in Rome, John had fun." becomes "OOV_1 arrived in OOV_2 yesterday. While in OOV_2, OOV_1 had fun." If a new unknown word x_i is encountered in $\mathbf{x}$ and all the available OOV tokens have been used, x_i is replaced by 'UNK', whose embedding is also updated during learning. The OOV tokens (and 'UNK') are included in the vocabulary, and SEQ^3 learns to predict them as summary words, in effect copying the corresponding unknown words of $\mathbf{x}$. At test time, we replace the OOV tokens with the corresponding unknown words.

A.3 Reconstruction Word Drop

Our model is an instance of Variational Auto-Encoders (VAE) (Kingma and Welling, 2014). A common problem in VAEs is that the reconstructor tends to disregard the latent variable. We weaken the reconstructor R, in order to force it to fully utilize the latent sequence $\mathbf{y}$ to generate $\hat{\mathbf{x}}$. To this end, we employ word dropout as in Bowman et al. (2016) and randomly drop a percentage of the input words, thus forcing R to rely solely on $\mathbf{y}$ to make good reconstructions.

A.4 Implementation and Hyper-parameters

We implemented SEQ^3 in PyTorch (Paszke et al., 2017). All the RNNs are LSTMs (Hochreiter and Schmidhuber, 1997). We use a shared encoder for the compressor and the reconstructor, consisting of a two-layer bidirectional LSTM with size 300 per direction. We use separate decoders for the compressor and the reconstructor; each decoder is a two-layer unidirectional LSTM with size 300. The (shared) embedding layer of the compressor and the reconstructor is initialized with 100-dimensional GloVe embeddings (Pennington et al., 2014) and is tied with the output (projec-

tion) layers of the decoders and jointly finetuned during training. We apply layer normalization (Ba et al., 2016) to the context vectors (Eq. 1) of the compressor and the reconstructor. We apply word dropout (§A.3) to the reconstructor with $p = 0.5$.

During training, the summary length M is sampled from $U(0.4\,N, 0.6\,N)$; during testing, $M = 0.5\,N$. The four losses are summed, λs being scalar hyper-parameters.

$$L = \lambda_R\, L_R + \lambda_P\, L_P + \lambda_T\, L_T + \lambda_L L_L$$

We set $\lambda_R = \lambda_T = 1$, $\lambda_L = \lambda_P = 0.1$. We use the Adam (Kingma and Ba, 2015) optimizer, with batch size 128 and the default learning rate 0.001. The network is trained for 5 epochs.

LM Prior The pretrained language model is a two-layer LSTM of size 1024 per layer. It uses its own embedding layer of size 256, which is randomly initialized and updated when training the language model. We apply dropout with $p = 0.2$ to the embedding layer and dropout with $p = 0.5$ to the LSTM layers. We use Adam (Kingma and Ba, 2015) with batch size 128 and the network is trained for 30 epochs. The learning rate is set initially to 0.001 and is multiplied with $\gamma = 0.5$ every 10 epochs.

Evaluation Following Chopra et al. (2016), we filter out pairs with empty headlines from the test set. We employ the PYROUGE package with "*-m -n 2 -w 1.2*" to compute ROUGE scores. We use the provided tokenizations of the Gigaword and DUC-2003, DUC-2004 datasets. All hyper-parameters were tuned on the development set.